

# Generalized Huffman Coding for Binary Trees with Choosable Edge Lengths

Jens Maßberg

*Institute of Optimization and Operations Research, University of Ulm, Germany,  
jens.massberg@uni-ulm.de*

---

## Abstract

In this paper we study binary trees with choosable edge lengths, in particular rooted binary trees with the property that the two edges leading from every non-leaf to its two children are assigned integral lengths  $l_1$  and  $l_2$  with  $l_1 + l_2 = k$  for a constant  $k \in \mathbb{N}$ . The depth of a leaf is the total length of the edges of the unique root-leaf-path.

We present a generalization of Huffman Coding that can decide in polynomial time if for given values  $d_1, \dots, d_n \in \mathbb{N}_{\geq 0}$  there exists a rooted binary tree with choosable edge lengths with  $n$  leaves having depths at most  $d_1, \dots, d_n$ .

*Keywords:* Combinatorial problems, Binary tree, Depth, Kraft's inequality, Huffman Coding, Choosable edge lengths

---

## 1. Introduction

For a fixed  $k \in \mathbb{N}$  we define  $\mathcal{L}'(k) = \{\{i, k-i\} \mid 1 \leq i \leq k-1, i \in \mathbb{N}\}$ . An  $\mathcal{L}'(k)$ -tree is a rooted strict binary tree with the property that the two edges leading from every non-leaf to its two children are assigned lengths  $l_1$  and  $l_2$  with  $\{l_1, l_2\} \in \mathcal{L}'(k)$ . In this context we will from now on call a multi-set  $D = \{d_1, \dots, d_n\}$  a *leaf signature*. In this paper we show for fixed  $k$  how to decide in polynomial time for a given leaf signature  $\{d_1, \dots, d_n\}$  if there exists an  $\mathcal{L}'(k)$ -tree with  $n$  leaves at depths at most  $d_1, \dots, d_n$  (in any order) and how to construct such a tree. See Figure 1 for an example of an  $\mathcal{L}'(6)$ -tree for the leaf signature  $\{5, 7, 7, 8, 8, 9\}$ . As the numbers of a leaf signature give an upper bound on the depths of the leaves, the tree of the example in Figure 1 is also an  $\mathcal{L}'(6)$ -tree for the leaf signatures  $\{7, 7, 7, 9, 9, 9\}$  and  $\{9, 9, 9, 9, 9, 9\}$ .

If  $k = 2$ , then all edges of an  $\mathcal{L}'(k)$ -tree have length 1. In this case we get classical binary trees where the depth of a leaf is equal to the number of edges of the unique path from the root to the leaf. By Kraft's inequality [8] an  $\mathcal{L}'(2)$ -tree for  $D$  exists if and only if

$$\sum_{i \in \{1, \dots, n\}} 2^{-d_i} \leq 1. \quad (1)$$

Such a tree can be constructed using the Huffman Coding algorithm [7].

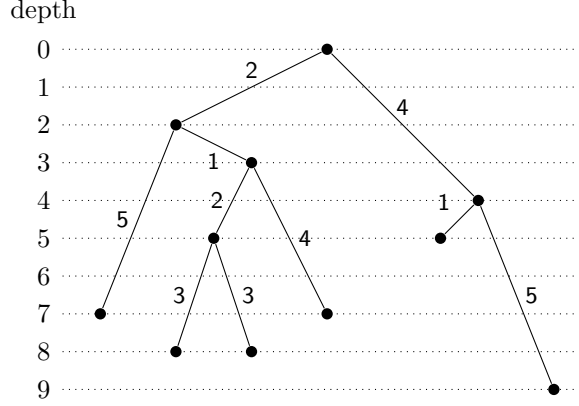


Figure 1: An  $\mathcal{L}'(6)$ -tree for the leaf signature  $\{5, 7, 7, 8, 8, 9\}$ .

$\mathcal{L}'(k)$ -trees have first been considered by Maßberg and Rautenbach [9]. They are motivated by the so-called repeater tree problem, a problem that occurs in VLSI-design. Repeater trees are tree structures consisting of wires and internal gates. They are required to distribute an electronical signal from a root to several locations on a chip, called sinks, while not exceeding individual time restrictions. In general there is a certain degree of freedom to distribute delay to different branches of the tree. By inserting a gate at a vertex of the tree it is possible to reduce the delay of one of the branches while increasing the delay on the other branch by about the same amount. As there are only a discrete number of gates with different sizes available, this effect can be modeled by the  $\mathcal{L}'(k)$ -trees: The lengths of the edges correspond to the delays on these edges and the  $d_i$  correspond to the required arrival times of a sink  $i$ . Then the task is to build an  $\mathcal{L}'(k)$ -tree such that the signal reaches each sink on time. For more details on the repeater tree problem and its connection to binary trees with choosable edge lengths we refer the reader to [1] and [6].

In [9] it has been shown how to construct  $\mathcal{L}'(4)$ -trees in polynomial time. For  $k > 4$  it was an open problem if there exists a polynomial algorithm that can decide the existence of  $\mathcal{L}'(k)$ -trees for given instances. We show that we can decide the existence in time  $O(n^{k+3})$ , that is, in polynomial time for constant  $k$ .

Our problem is also related to prefix-free codes with unequal letter costs (see e.g. [2, 3, 4, 5]). Nevertheless, there are significant differences between these problems. First, in our problem we ask for a tree satisfying given depth restrictions for the leaves while for prefix-free codes the task is to find a tree minimizing  $\sum_{i \in \{1, \dots, n\}} \text{depth}(i) w_i$  where  $\{w_1, \dots, w_n\}$  are given numbers and  $\text{depth}(i)$  denotes the depth of leaf  $i$ ,  $i \in \{1, \dots, n\}$ . Moreover, in our problem we have the freedom to choose the edge lengths from a discrete set of numbers

as long as the sum of the lengths of the edges leaving a vertex equals  $k$ .

## 2. Core Algorithm

Let  $D = \{d_1, \dots, d_n\}$  be a leaf signature. We want to decide if there exists an  $\mathcal{L}'(k)$ -tree with  $n$  leaves at depths at most  $d_1, \dots, d_n$ . If such a tree exists we call the leaf signature  $D$  *realizable*.

In this section we present the core algorithm that can decide the existence of an  $\mathcal{L}'(k)$ -tree for a given leaf signature. The idea of the algorithm is the following: We can reduce a leaf signature of length  $n$  by combining two leaves of a potential tree into one slightly higher up in the tree. This leads to a set of leaf signatures of length  $n - 1$ . If one of them is realizable, the original one is realizable, too. By iterative application of this method we get a set of leaf signatures of length 1 where the existence of an  $\mathcal{L}'(k)$ -tree can be checked easily. In Section 3 we refine the algorithm to get a polynomial running time.

For two integers  $a_1, a_2 \in \mathbb{Z}$  we define

$$\omega_k(a_1, a_2) := \min\{a_1, a_2\} - \max\left\{1, \left\lceil \frac{k - |a_1 - a_2|}{2} \right\rceil\right\}. \quad (2)$$

The idea of the core algorithm relies on the following observation.

**Proposition 2.1.** *A leaf signature  $D = \{d_1, \dots, d_n\}$ ,  $n \geq 2$ , is realizable if and only if there exist  $i, j$  with  $1 \leq i < j \leq n$  such that  $D \setminus \{d_i, d_j\} \cup \{\omega_k(d_i, d_j)\}$  is realizable.*

*Proof:* Assume that  $D$  is realizable and let  $T$  be an  $\mathcal{L}'(k)$ -tree realizing  $D$ . Then there must be two leaves  $v$  and  $w$  of  $T$  that have a common parent  $u$ . Let  $d_i$  and  $d_j$ ,  $i < j$ , be the depth limits from  $D$  assigned to  $v$  and  $w$ .

Let  $d : V(T) \rightarrow \mathbb{N}$  be the depth of the vertices of  $T$  with respect to the length of the edges. We show that  $d(u) \leq \omega_k(d_i, d_j)$ . As  $T$  is an  $\mathcal{L}'(k)$ -tree realizing  $D$  we have  $d(u) \leq d(v) - 1 \leq d_i - 1$ ,  $d(u) \leq d(w) - 1 \leq d_j - 1$  and  $k = (d(v) - d(u)) + (d(w) - d(u)) \leq d_i + d_j - 2d(u)$ . We conclude, using that  $d(u) \in \mathbb{N}$ :

$$d(u) \leq \min\{d_i, d_j\} - 1$$

and

$$d(u) \leq \left\lfloor \frac{d(v) + d(w) - k}{2} \right\rfloor = \min\{d_i, d_j\} - \left\lceil \frac{k - |d_i - d_j|}{2} \right\rceil.$$

Therefore we have  $d(u) \leq \omega_k(d_i, d_j)$  and the tree  $(V(T) \setminus \{v, w\}, E(T) \setminus \{uv, uw\})$  realizes  $D \setminus \{d_i, d_j\} \cup \{\omega_k(d_i, d_j)\}$ .

On the other hand assume, that there are  $i, j \in \{1, \dots, n\}$ ,  $i < j$ , such that  $D' = (D \setminus \{d_i, d_j\}) \cup \{\omega_k(d_i, d_j)\}$  is realizable. Let  $T$  be an  $\mathcal{L}'(k)$ -tree realizing  $D'$  and let  $v$  be the leaf of  $T$  assigned to  $\omega_k(d_i, d_j)$ . Add two edges of length  $a = \max\{k - 1, d_i - \omega_k(d_i, d_j)\}$  and  $b = k - a$  at  $v$ . Then the two newly inserted leaves have depth at most  $d_i$  and  $d_j$  and thus the new tree realizes  $D$ .  $\square$

By repeatedly applying Prop. 2.1 we get to the algorithm outlined at the beginning of the section. Unfortunately, the number of leaf signatures that are computed grows exponentially in the length of the initial leaf signature.

### 3. Refined Algorithm

In this section we refine the algorithm outlined in the previous section in order to get a polynomial running time. The refinement depends on the observations that the values of a leaf signature cannot be too big and that we can restrict ourselves to leaf signatures identical to the original signature except for the bottom  $k$  layers.

First we introduce the notion of domination which will be useful in the proofs.

**Definition 3.1.** *Let  $A = \{a_1, \dots, a_n\}$  and  $B = \{b_1, \dots, b_n\}$  be two leaf signatures of the same length. If there exists a permutation  $\pi : \{1, \dots, n\} \rightarrow \{1, \dots, n\}$  such that  $a_i \leq b_{\pi(i)}$  for all  $i \in \{1, \dots, n\}$  then  $A$  is dominated by  $B$ .*

Obviously, it is sufficient to consider only sets of leaf signatures, where no signature is dominated by another.

Next we note that we can assume the values of a leaf signature not to be too big.

**Proposition 3.2.** *The leaf signature  $D = \{d_1, \dots, d_n\}$  is realizable if and only if the leaf signature  $D' = \{\min\{d_1, (k-1)(n-1)\}, \dots, \min\{d_n, (k-1)(n-1)\}\}$  is realizable.*

*Proof:* Assume there is an  $\mathcal{L}'(k)$ -tree  $T$  realizing  $D$ . As  $T$  contains  $n$  leaves, every root-leaf-path consists of at most  $n-1$  edges. Moreover, in an  $\mathcal{L}'(k)$ -tree every edge has length at most  $k-1$ . We conclude that the leaves of  $T$  have depth at most  $\min\{d_1, (k-1)(n-1)\}, \dots, \min\{d_n, (k-1)(n-1)\}$  and therefore  $D'$  is realizable.

The other direction of the proof follows from the fact, that  $D'$  is dominated by  $D$ .  $\square$

Now we show that we can assume each computed leaf signature to be identical to the original signature except for the bottom  $k$  layers.

**Proposition 3.3.** *The leaf signature  $D = \{d_1, \dots, d_n\}$  is realizable if and only if there exist  $i, j$  with  $1 \leq i < j \leq n$  such that the signature*

$$(\{\min(d_1, \omega), \dots, \min(d_n, \omega)\} \setminus \{\min(d_i, \omega), \min(d_j, \omega)\}) \cup \{\omega_k(d_i, d_j)\} \quad (3)$$

*with  $\omega = \omega_k(d_i, d_j) + k - 1$  is realizable.*

*Proof:* Let  $T$  be an  $\mathcal{L}'(k)$ -tree realizing  $D$ , let  $u$  be an internal vertex of  $T$  of maximum depth and denote by  $v, w$  the two children of  $u$ . Obviously,  $v$  and  $w$  are leaves of  $T$ . Let  $d_i$  and  $d_j$ ,  $1 \leq i < j \leq n$ , be the values assigned to  $v$  and  $w$ . By the proof of Prop. 2.1 we have  $d(u) \leq \omega_k(d_i, d_j)$ . As  $u$  is an internal vertex

of maximum depth and all edges in  $T$  have length at most  $k - 1$ , all leaves of  $T$  have depth of at most  $d(u) + k - 1 \leq \omega_k(d_i, d_j) + k - 1$ . Deleting the leaves  $v, w$  and their incident edges from  $T$  we obtain a tree realizing the signature (3).

On the other hand assume that there are  $i$  and  $j$  with  $1 \leq i < j \leq n$  such that  $A = (\{\min(d_1, \omega), \dots, \min(d_n, \omega)\} \setminus \{\min(d_i, \omega), \min(d_j, \omega)\}) \cup \{\omega_k(d_i, d_j)\}$  is realizable for  $\omega = \omega_k(d_i, d_j) + k - 1$ . Note, that  $A$  is dominated by  $A' = (\{d_1, \dots, d_n\} \setminus \{d_i, d_j\}) \cup \{\omega_k(d_i, d_j)\}$  and thus  $A'$  is realizable. Then by Proposition 2.1 the leaf signature  $D$  is realizable.  $\square$

The above results lead us to Algorithm 1 where one after another set  $\mathcal{M}_z$  of leaf signatures of length  $z$  are computed, starting with  $\mathcal{M}_n$  containing only the initial leaf signature of length  $n$ .

**Input:** An instance  $D = \{d_1, \dots, d_n\}$  and  $k \in \mathbb{N}, k \geq 2$ .

**Output:** Returns true iff there exists an  $\mathcal{L}'(k)$ -tree for  $D$ .

```

1  $\mathcal{M}_n \leftarrow \{D\};$ 
2 for  $z = n - 1$  to 1 do
3    $\mathcal{M}_z \leftarrow \emptyset;$ 
4   foreach leaf signature  $A = \{a_1, \dots, a_{z+1}\} \in \mathcal{M}_{z+1}$  do
5      $\mathcal{B}_A \leftarrow \emptyset;$ 
6     foreach  $i, j \in \{1, \dots, z + 1\}, i < j$  do
7       Compute leaf signature  $B$  out of  $A$  by replacing  $a_i$  and  $a_j$  by
        $\omega_k(a_i, a_j)$  and truncating large values in  $B$  according to Prop.
       3.2 and Prop. 3.3;
8        $\mathcal{B}_A \leftarrow \mathcal{B}_A \cup \{B\};$ 
9     end
10    Remove all signatures from  $\mathcal{B}_A$  that are dominated by other
    signatures in  $\mathcal{B}_A$ ;
11     $\mathcal{M}_z \leftarrow \mathcal{M}_z \cup \mathcal{B}_A;$ 
12  end
13  (optional: Remove leaf signatures  $A \in \mathcal{M}_z$  that are dominated by
  other leaf signatures  $B \in \mathcal{M}_z$ );
14 end
15 if  $\{i\} \in \mathcal{M}_1$  for some  $i \geq 0$  then return true;
16 return false;
```

**Algorithm 1:**  $\mathcal{L}'(k)$ -tree decision algorithm.

Figure 2 shows the sets  $\mathcal{M}_z$ ,  $z \in \{1, \dots, 6\}$ , computed by the algorithm for the instance  $\{5, 7, 7, 8, 8, 9\}$ . Dominated signatures are removed. The arrows show where each new signature comes from. As the final set  $\mathcal{M}_0$  contains the leaf signature  $\{0\}$ , the initial signature is realizable.

Let  $D = \{d_1, \dots, d_n\}$  be the input of the algorithm. In order to be able to reproduce the reduction steps we introduce a function  $p(\cdot)$  that assigns a leaf signature  $B$  to the signature it replaces, that is,  $p(B) = A$  for  $A$  and  $B$  as in Line 7 of the algorithm. For a leaf signature  $B$  we denote by  $l(B)$  the minimum value of an element that has been added to  $B$ , that is,  $l(D) = \infty$  and

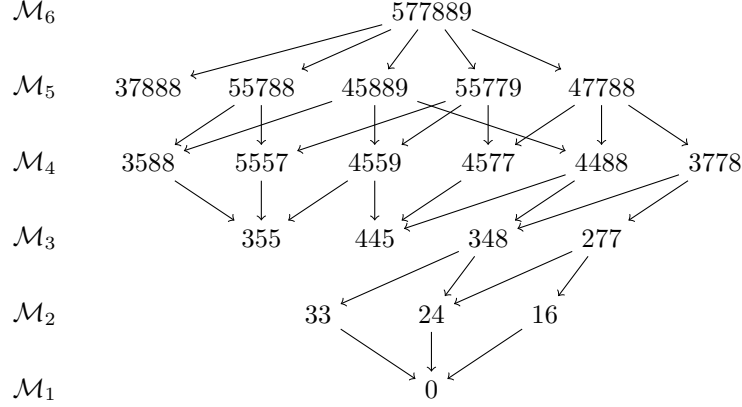


Figure 2: Computed leaf signatures for the instance  $\{5, 7, 7, 8, 8, 9\}$ .

$l(B) = \min\{l(A), \omega(a_i, a_j)\}$  for  $A, B, i, j$  as in Line 7. This implies that we have only removed and changed elements  $b$  with  $b > l(B)$ . Thus if  $B = \{b_1, \dots, b_z\}$  with  $b_1 \leq \dots \leq b_z$  and  $d_1 \leq \dots \leq d_z$  then

$$b_i = d_i \text{ for all } i \leq \max\{j \mid b_j < l(B)\} = \max\{j \mid d_j < l(B)\}. \quad (4)$$

The algorithm computes all necessary leaf signatures according to Prop. 3.2 and Prop. 3.3. Consequently, a leaf signature in  $\mathcal{M}_z$  is realizable if and only if a leaf signature in  $\mathcal{M}_{z+1}$  is realizable for all  $z \in \{1, \dots, n-1\}$  proving the correctness of the algorithm.

For simplicity of notation we set  $m(A) = \max_{a \in A} a$ . We will prove now that the running time of Algorithm 1 is polynomially bounded in  $n$  for fixed  $k$ . To this end we show that the sizes of the sets  $\mathcal{M}_z$  are polynomially bounded in  $n$ . First we show that for every computed leaf signature  $B$  the largest element in  $B$  is at most  $k-1$  larger than the smallest element that has been inserted into  $B$  by the algorithm.

**Proposition 3.4.** *If  $z \in \{1, \dots, n-1\}$ , then all leaf signatures  $B \in \mathcal{M}_z$  satisfy*

$$m(B) \leq l(B) + k - 1. \quad (5)$$

*Proof:* For contradiction we assume that  $B$  is a set of maximum cardinality computed by the algorithm that contradicts (5).

Set  $A = p(B)$ . Obviously,  $l(B) \leq l(A)$  and  $m(B) \leq m(A)$ . If  $l(B) = l(A)$  then  $m(A) \geq m(B) > l(B) + k - 1 = l(A) + k - 1$ , i.e.  $A$  also does not satisfy (5), contradicting the maximality of  $B$ .

Thus  $l(B) < l(A)$ . But in this case  $l(B) = \omega(a_i, a_j)$  for the two elements  $a_i, a_j \in A$  that are replaced. As all values of  $B$  are truncated in Line 7 of the algorithm according to Prop. 3.3, we obtain  $m(B) \leq m \leq \omega(a_i, a_j) + k - 1 = l(B) + k - 1$ , which is a contradiction and completes the proof.  $\square$

Using the previous result we show that the size of  $\mathcal{M}_z$  is polynomially bounded in  $z$ .

**Lemma 3.5.** *If  $z \in \{1, \dots, n-1\}$ , then*

$$|\mathcal{M}_z| \leq z^k. \quad (6)$$

*Proof:* Define  $\mathcal{M}_z^i := \{A \in \mathcal{M}_z : l(A) = i\}$  for  $i \in \mathbb{N}$ . By Prop. 3.2 and the truncation of large values in Line 7 of the algorithm according to Prop. 3.2, we know  $\mathcal{M}_z^i = \emptyset$  for  $i > (z-1)(k-1)$ .

Let  $i \in \{0, \dots, (z-1)(k-1)\}$  and  $A = \{a_1, \dots, a_z\} \in \mathcal{M}_z^i$  such that  $a_1 \leq \dots \leq a_z$ . Recall that  $D = \{d_1, \dots, d_n\}$ ,  $d_1 \leq \dots \leq d_n$ , is the input of the algorithm. Set  $t = \max\{j : d_j < l(A)\}$ . By the definition of  $l(A)$  and (4), we conclude

$$a_j = d_j \quad (7)$$

for  $j \in \{1, \dots, t\}$ . On the other hand, by Prop. 3.4,

$$a_j \in \{l(A), \dots, l(A) + k - 1\} \quad (8)$$

for  $j \in \{t+1, \dots, z\}$ .

This implies that the sets  $A \in \mathcal{M}_z^i$  only differ in the largest  $(z-t)$  elements and each of these elements can only take one of  $k$  different values. Thus the number of different sets  $A$  (without removing dominated ones) is at most the number of integral partitions of  $z-t$  into the sum of  $k$  non-negative integers. This number equals  $\binom{z-t+k-1}{k-1}$  and is bounded by  $\frac{(z-t)^{k-1}}{k-1} \leq \frac{z^{k-1}}{k-1}$ .

Altogether we have at most  $1 + (z-1)(k-1)$  sets  $\mathcal{M}_z^i$  that are non-empty and each of these sets has at most  $z^{k-1}$  elements. Hence

$$|\mathcal{M}_z| = \sum_{i \in \mathbb{N}} |\mathcal{M}_z^i| \leq (1 + (z-1)(k-1)) \frac{z^{k-1}}{k-1} \leq z^k. \quad (9)$$

This finishes the proof.  $\square$

Before we can prove the running time of the algorithm we show that for each leaf signature  $A$  the size of the set  $\mathcal{B}_A$  is not too big.

**Lemma 3.6.** *For any leaf signature  $A = \{a_1, \dots, a_{z+1}\}$  the set  $\mathcal{B}_A$  contains at most  $k \cdot z$  elements. Moreover, the set  $\mathcal{B}_A$  can be computed in time  $O(kn^2)$ .*

*Proof:* W.l.o.g. assume  $a_1 \leq \dots \leq a_{z+1}$ . For every pair  $i, j \in \{1, \dots, z+1\}$ ,  $i \leq j$ , a new signature  $B$  is computed out of  $A$  in the **foreach** loop in Line 6. We denote this signature by  $B_{i,j}$ .

First note that  $\omega_k(a_i, a_j) = a_i - \max\{1, \lceil (k - a_j + a_i)/2 \rceil\}$  and thus

$$a_i - k + 1 \leq a_i - \lceil k/2 \rceil \leq \omega_k(a_i, a_j) \leq a_i - 1. \quad (10)$$

Now assume that there exist indices  $i, j, j'$ ,  $1 \leq i \leq j \leq j' \leq z+1$ , such that  $\omega = \omega_k(a_i, a_j) = \omega_k(a_i, a_{j'})$ . Then  $B_{i,j} = (B_{i,j'} \setminus \{\min\{a_{j'}, (z-1)(k-1), \omega +$

$k-1\}\}) \cup \{\min\{a_j, (z-1)(k-1), \omega+k-1\}\}$ , that is,  $B_{i,j}$  is dominated by  $B_{i,j'}$ . We conclude that the maximum number of signatures in  $\mathcal{B}_A$  depends on the number of possible values for  $a_i$  and  $\omega$ , respectively.

Instead of traversing all values for  $i$  and  $j$  in order to compute  $\mathcal{B}_A$  it suffices to traverse all possible values of  $i$  and  $\omega$ . In order to construct a new signature for given  $i$  and  $\omega$  we have to traverse  $z+1$  elements, implying the total running time of  $O(kz^2)$ .  $\square$

Joining the previous results together we are able to prove that the running time is polynomially bounded in the size of the input.

**Theorem 3.7.** *For a given leaf signature  $D = \{d_1, \dots, d_n\}$  it can be decided in time  $O(n^{k+3})$  if there exists an  $\mathcal{L}'(k)$ -tree with  $n$  leaves at depth at most  $d_1, \dots, d_n$ . Moreover, such a tree can be constructed in the same running time.*

*Proof:* To achieve this running time we combine Algorithm 1 and the idea of Lemma 3.6 in order to compute the sets  $\mathcal{B}_A$ . By Lemma 3.6 each set  $\mathcal{B}_A$  can be computed in time  $O(kz^2)$  for  $z+1 = |A|$ . Applying Lemma 3.5 the total running time is bounded by

$$O\left(\sum_{z \in \{1, \dots, n-1\}} kz^2 |\mathcal{M}_z|\right) \subseteq O\left(\sum_{z \in \{1, \dots, n-1\}} kz^{k+2}\right) \subseteq O(n^{k+3}). \quad (11)$$

As we have seen before, there exists an  $\mathcal{L}'(k)$ -tree realizing the signature  $D$  if and only if  $\{i\} \in \mathcal{M}_1$  for an  $i \geq 0$ . This tree can be constructed using the predecessor function  $p(\cdot)$ .  $\square$

In practice the running time can be improved by removing dominated signatures.

**Remark 3.8.** *By removing dominated signatures after each iteration of the main loop (see Line 13 of Alg. 1), the cardinalities of the sets  $\mathcal{M}_z$  and the running time of the algorithm can be reduced significantly in practice. Nevertheless, it seems that the theoretical worst case running times do not decrease in general.*

#### 4. Conclusion and Future Work

In this paper we have presented an algorithm building rooted binary trees with choosable edge lengths in polynomial time for fixed  $k$ . This algorithm can be seen as a generalization of Huffman coding: If  $k = 2$  we are in the case of ordinary binary trees with all edges having length 1. In this case it is easy to show that for any leaf signature  $A = \{a_1, \dots, a_z\}$  the set  $\mathcal{B}_A$  only contains the signature we get by replacing the largest two elements  $a_i, a_j$ ,  $1 \leq i < j < z$ , by  $\omega_2(a_i, a_j) = \min\{a_i, a_j\} - 1$  (after removing dominated signatures). Thus  $|\mathcal{M}_z| = 1$  for all  $z \in \{1, \dots, n\}$  and the algorithm is equivalent to Huffman coding.

It is still open if there is an algorithm for the  $\mathcal{L}'(k)$ -tree with a significantly better running time, for example an algorithm with a running time that is polynomially bounded not only in  $n$  but also in  $k$ .



## References

- [1] C. Bartoschek, S. Held, J. Maßberg, D. Rautenbach, and J. Vygen. The repeater tree construction problem. *Information Processing Letters*, 110:1079–1083, 2010.
- [2] P. Bradford, M. Golin, L. L. Larmore, and W. Rytter. Optimal prefix-free codes for unequal letter costs and dynamic programming with the monge property. *Journal of Algorithms*, 42(2):277–303, 2002.
- [3] E. N. Gilbert. Coding with digits of unequal cost. *IEEE Transactions on Information Theory*, 41(2):596–600, 1995.
- [4] M. Golin, C. Mathieu, and N. E. Young. Huffman coding with letter costs: A linear-time approximation scheme. *SIAM Journal on Computation*, 41(3):684–713, 2012.
- [5] M. Golin and G. Rote. A dynamic programming algorithm for constructing optimal prefix-free codes for unequal letter costs. *IEEE Transactions on Information Theory*, 44(5):1770–1781, 1998.
- [6] S. Held and S. Rotter. Shallow-light Steiner arborescences with vertex delays. *Integer Programming and Combinatorial Optimization*, pages 229–241, 2013.
- [7] D.A. Huffman. A method for the construction of minimum-redundancy codes. In *Proceedings of the IRE*, volume 40, pages 1098–1101, 1952.
- [8] L.G. Kraft. A device for quantizing, grouping, and coding amplitude modulated pulses. Master’s thesis, MIT, Cambridge, 1949.
- [9] J. Maßberg and D. Rautenbach. Binary trees with choosable edge lengths. *Information Processing Letters*, 109(18):1087–1092, 2009.